OMSC Army Platform Abstract Object Class

(Revised: 1 June 1999)

The OMSC designed a Platform Object Class for use in Army models and simulations (M&S). The development of this class is based on a component approach to class development. This component approach supports the use of both object composition and class inheritance methodologies. The resulting class is composable in nature. This allows each M&S developer to tailor the object to their specific application while maintaining a standard model structure and a minimum set of common model interfaces. The Platform Object is organized along the following lines:

- Platform Superclass
  - Platform Component Class
    - Sensor Class
    - Weapon Class
    - Platform Frame Class
      - Frame Component Subclass
    - Movement Class
    - Logistics Class
      - Supply Subclass
      - Maintenance Subclass
    - Crew Class
    - Communication Class
    - Carrier Class

The attached figure outlines the structure of the Platform Object. Also provided are definitions for each of the components of the Platform Object class as well as descriptions of the public methods associated with each class.

Platform Object Class and Component Definitions

Class Platform. A platform can be any entity of interest in the model. Examples include vehicles of all types, individuals/persons, individual systems (i.e., radar systems), a missile, etc.
Public Methods:

- getType(): Returns the type designation for the platform.
- getStatus(): Returns the platform status. The status is typically an enumeration of the standard kill categories (M, F, MF, or K). It can simply be either alive/dead (1/0). It can be derived from the component status.
- getLocation(): Returns the current platform location.
- getSide(): Returns the faction or coalition for the platform. There is no implied enmity between sides.
- assessDamage(): Used to instruct the platform to calculate the damage caused by another object.

Class PlatformComponent. A platform is partitioned into logical components so that the modeler can compose a platform from the components. Components may be extended through inheritance. All of the components listed below will inherit the following two methods from this class.
Public Methods:

- getType(): Returns the component type designation.
- getStatus(): Returns the status of a component; status is typically either functional or nonfunctional (1/0).

Class Sensor. This models the component of a platform that detects other platforms. Examples of sensors include crew vision, infrared sights, and radar.
Public Methods:

- getMaxRange(): Returns the maximum range of the sensor (may be used to reduce the area to be searched).
- getOrientation(): Returns the direction of sensor orientation.
- getContacts(): Used to query the targets currently visible to the sensor component.
- activate(): Used to place the sensor in an active mode.
- deactivate(): Used to place the sensor in an active mode.

Class Weapon. Used to describe the weapon systems on the platform.
Public Methods:

- getMaxRange(): Return the max range for specified munition.
- load(): Used to load a munition (this creates the weapon/munition pair).
- engageTarget(): Used to initiate the weapon-firing event.

Platform Object Class and Component Definitions
(continued)

Class PlatformFrame. The component contains the physical description of the platform. This may be a detailed model, but typically is data required by sensors to acquire/detect the platform. Examples of the physical data are the visual signature, thermal signature, acoustic signature and cross sectional area. Platform orientation and other descriptions also belong here.

Public Methods:

getSignature(): Returns the signature of the target appropriate for the type of sensor being used.

Class FrameComponent. FrameComponents can be used to describe individual parts of the PlatformFrame. Providing separate descriptions for both the hull and turret of a tank is one use of this component.

Public Methods:

getSignature(): Returns the signature of the target component appropriate for the type of sensor being used.

Class Movement. This class describes the movement capabilities of a platform.

Public Methods:

getVelocity(): Returns the current velocity (direction of movement and rate) of the platform.
changeVelocity(): Used to request a change in velocity.
moveTo (): Used to order the platform to move directly to a location.

Class Logistics. This component is intended to capture or represent the internal logistics capability and/or requirements of the platform.

Public Methods:

receive(): Used to increment the quantity of this logistic component.

Class Supply. This component is intended to represent individual classes of supply used by the platform. Ammunition could be one example of this class.

Public Methods:

getRemainingCapacity(): Returns the remaining capacity for this supply component.
getTotalCapacity(): Returns the total capacity for this supply component.
getQuantityOnHand(): Returns the quantity of this supply that is on hand.
expend(): Used to expend a quantity of the supply component.
transfer(): Used to transfer a quantity of an on hand supply component to another platform.

Platform Object Class and Component Definitions
(continued)

Class Maintenance. This component is intended to represent maintenance actions/requirements of the platform. Since the platform object can be used to describe both systems and people the action can also be used to describe the medical treatment of injuries.
Public Methods:
conduct_maintenance(): Used to perform maintenance action on platform.

Class Crew. This component is intended to represent individual crew activities for a platform.
Public Methods:
getQuantity(): Returns the number of crewmembers on the platform.

Class Communications. Provides the platform the ability to send and receive messages.
Public Methods:
getNet(): Returns the collection of objects capable of exchanging messages.
getNet(): Used to add the platform to the collection of objects capable of exchanging messages.
sendMessage(): Used to send a message on the net.
receiveMessage(): Used to receive a message from the net.

Class Carrier. This component allows the platform to carry other objects. Examples of items that could be carried include other platforms, individuals (i.e., non-crew), and supplies.
Public Methods:
load(): Used to load objects on the carrier.
unload(): Used to unload objects carried.
getRemainingCapacity(): Return the number of additional objects of this type that can be loaded.
getTotalCapacity(): Return the total number of objects of this type that can be carried.
getQtyOnHand(): Returns the number of this type on hand.